

# 1. Antizadatak - zadatak iz Logo-a

## Funkcije:

`iks(a)`

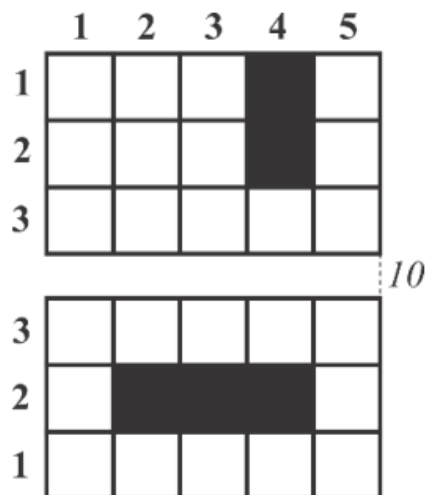
- Funkcija koja crta X, odnosno dijagonale kvadrata.
- Ulazni podatak je duljina stranice kvadrata.
- Funkcija nema izlaz. Pri pozivu crta dijagonale kvadrata (u našem slučaju dijagonale jednog polja ploče).

`kvadrat(a)`

- Funkcija crta kvadrat s duljinom stranice a.
- Ulazni podatak je duljina stranice kvadrata.
- Funkcija nema izlaz. Pri pozivu crta kvadrat (u našem slučaju jedno polje ploče).

`mreza(n, m, d)`

- Funkcija koja crta ploču za igru potapanje brodova. Ploča je oblika kako je zadano u zadatku.



- Ulazni podaci su:
  - `n` broj kvadrata (polja) u retku
  - `m` broj kvadrata (polja) u stupcu
  - `d` duljina stranice kvadrata (polja)
- Funkcija nema izlaz. Pri pozivu funkcije, crtaju se dvije "ploče".

`ucrtaj_brodove(tomek, medo, n, m, d)`

- Funkcija na ploču ucrtava Tomekove i Medine brodove. Ova funkcija koristi funkciju `mreza(n, m, d)`.
- Ulazni podaci su:

<code>tomek</code>	lista Tomekovih brodova
<code>medo</code>	lista Medinih brodova
<code>n</code>	broj kvadrata (polja) u retku
<code>m</code>	broj kvadrata (polja) u stupcu
<code>d</code>	duljina stranice kvadrata (polja)
- Funkcija vraća 'Smjer nije dobar' ukoliko ulazni podaci nisu dobri, inače nema izlaz.
- Sami moramo paziti u koja polja ćemo ucrtati brodove. Funkcije će raditi za sve upisane brojeve, ali brodovi neće biti ucrtani na dobra mjesta. Dakle, može se dogoditi da su brodovi ucrtani izvan polja za igru.

`provjera(l, tomek, medo)`

- Funkcija provjerava nalaze li se odabrana polja (pokušaji gađanja polja) iz liste `l` među poljima na kojima se nalaze brodovi Tomeka i Mede.
- Funkcija uspoređuje brodove iz liste `l` i koordinate Tomekovih i Medinih lista te njihove duljine. U slučaju promašaja prebacuje se na drugog igrača i nastavlja provjeravati njegove brodove.
- Ulazni podaci su:

<code>tomek</code>	lista pozicija Tomekovih brodova
<code>medo</code>	lista pozicija Medinih brodova
<code>l</code>	lista koja sadrži podliste oblika <code>[r s]</code> , pri čemu <code>r</code> i <code>s</code> predstavljaju redak i stupac unutar protivničkog polja za koje je igrač koji je na potezu postavio pitanje
- Funkcija vraća listu koja sadrži dvije liste, brodove koje je pogodio Tomek te brodove koje je pogodio Medo `[pog_t, pog_m]`.

`brodovi(n, m, d, tomek, medo, l)`

- Funkcija briše ucrtane brodove i ucrtava X-eve za one brodove koji su pogođeni. Ova funkcija koristi funkcije `provjera(l, tomek, medo)` i `ucrtaj_brodove(tomek, medo, n, m, d)`.

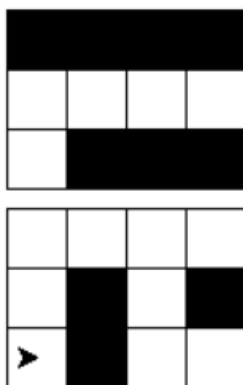
- Ulazni podaci su:  $n$  broj kvadrata (polja) u retku  
 $m$  broj kvadrata (polja) u stupcu  
 $d$  duljina stranice kvadrata (polja)  
 $tomek$  lista pozicija Tomekovih brodova  
 $medo$  lista pozicija Medinih brodova  
 $l$  lista koja sadrži podliste oblika  $[r \ s]$ , pri čemu  $r$  i  $s$  predstavljaju redak i stupac unutar protivničkog polja za koje je igrač koji je na potezu postavio pitanje
- Funkcija nema izlaz.
- Ukoliko su u listama upisani brojevi izvan granica ploče, moguće je da se i brodovi i “promašaji” i “pogoci” ucrtaju izvan ploče.

### Testni primjeri:

Ulaz:

```
brodovi(3, 4, 30, [[1,2,'O',2], [2,4,'V',1]],
[[1,1,'V',4],[3,2,'V',3]], [])
```

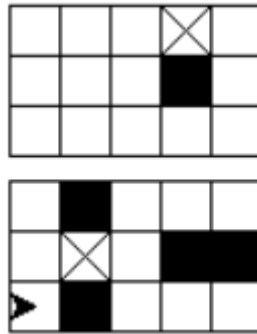
Izlaz:



Ulaz:

```
brodovi(3, 5, 20, [[1,2,'O',3], [2,4,'V',2]],
[[2,4,'O',2]], [[1,4],[2,3],[2,2]])
```

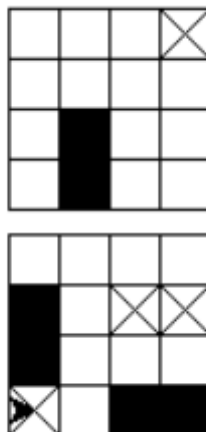
Izlaz:



Ulaz:

```
brodovi(4, 4, 20, [[1,1,'O',3], [1,3,'V',2],  
[3,3,'V',2]], [[4,2,'O',2], [1,4,'O',1]],  
[[1,4],[2,4],[1,1],[3,4],[3,3]])
```

Izlaz:



Kod:

```
import turtle
import math

s = turtle.getscreen()
t = turtle.getturtle()
t.speed(20)

def iks(a):
    dijagonala = a * math.sqrt(2)
```

```
t.penup()
t.forward(a / 2)
t.left(90)
t.forward(a / 2)
t.right(45)
t.pendown()
t.back(dijagonala)
t.penup()
t.right(45)
t.forward(a)
t.right(45)
t.pendown()
t.back(dijagonala)
t.penup()
t.right(45)
t.forward(a / 2)
t.left(90)
t.forward(a / 2)
return
```

```
def kvadrat(a):
    t.penup()
    t.forward(a / 2)
    t.pendown()
    t.right(90)
    for i in range(2):
        t.forward(a / 2)
        t.right(90)
        t.forward(a)
        t.right(90)
        t.forward(a / 2)
    t.left(90)
    t.penup()
    t.back(a / 2)
    t.pendown()
    return
```

```
def mreza(n, m, d):
```

#n je broj redaka, m je broj stupaca, d je  
duljina stranice kvadrata

```

for k in range(2):
    for j in range(n):
        for i in range(m):
            kvadrat(d)
            t.penup()
            t.forward(d)
            t.pendown()
        t.penup()
        t.back(m * d)
        t.left(90)
        t.forward(d)
        t.right(90)
        t.pendown()
    t.penup()
    t.left(90)
    t.forward(10)
    t.right(90)
    t.pendown()
t.penup()
t.left(90)
t.back(2 * 10 + 2 * n * d)
t.right(90)
t.pendown()
return

```

```

def ucrtaj_brodove(tomek, medo, n, m, d):
    mreza(n, m, d)
    t.penup()
    t.left(90)
    for lista in tomek:
        t.forward((lista[0] - 1) * d)
        t.right(90)
        t.forward((lista[1] - 1) * d)
        t.begin_fill()
        kvadrat(d)
        t.end_fill()

```

```

t.penup()
if lista[2] == 'V':
    for i in range(lista[3] - 1):
        t.forward(d)
        t.begin_fill()
        kvadrat(d)
        t.end_fill()
        t.penup()
    t.back((lista[3] - 1) * d)
elif lista[2] == 'O':
    t.left(90)
    for i in range(lista[3] - 1):
        t.forward(d)
        t.begin_fill()
        kvadrat(d)
        t.end_fill()
        t.penup()
    t.back((lista[3] - 1) * d)
    t.right(90)
else:
    return 'Smjer nije dobar'
t.back((lista[1] - 1) * d)
t.left(90)
t.back((lista[0] - 1) * d)
#pos = t.setpos()
t.forward((2 * n - 1) * d + 10)
t.right(180)
for lista in medo:
    t.forward((lista[0] - 1) * d)
    t.left(90)
    t.forward((lista[1] - 1) * d)
    t.begin_fill()
    kvadrat(d)
    t.end_fill()
    t.penup()
    if lista[2] == 'V':
        for i in range(lista[3] - 1):
            t.forward(d)

```

```

        t.begin_fill()
        kvadrat(d)
        t.end_fill()
        t.penup()
        t.back((lista[3] - 1) * d)
    elif lista[2] == 'O':
        t.left(90)
        for i in range(lista[3] - 1):
            t.forward(d)
            t.begin_fill()
            kvadrat(d)
            t.end_fill()
            t.penup()
        t.back((lista[3] - 1) * d)
        t.right(90)
    else:
        return 'Smjer nije dobar'
    t.back((lista[1] - 1) * d)
    t.right(90)
    t.back((lista[0] - 1) * d)

    t.right(180)
    t.back((2 * n - 1) * d + 10)
    t.right(90)
    return

def provjera(l, torek, medo):
    pog_t = []
    pog_m = []
    pogodak = 0
    red = "T" # ili "T" ili "M"
    for lista in l:
        pogodak = 0
        if red == "T":
            for lista2 in medo:
                if lista2[2] == "O":

```



```

        if (lista[1] == lista2[1]) and
(lista[0] <= lista2[0]) and (lista[0] >= lista2[0] -
lista2[3] + 1):
            pog_t.append(lista)
            pogodak = 1
        else:
            if (lista[0] == lista2[0]) and
(lista[1] >= lista2[1]) and (lista[1] <= lista2[1] +
lista2[3] - 1):
                pog_t.append(lista)
                pogodak = 1
            if pogodak == 0:
                red = "M"
            if red == "M":
                for lista2 in tomek:
                    if lista2[2] == "O":
                        if (lista[1] == lista2[1]) and
(lista[0] >= lista2[0]) and (lista[0] <= lista2[0] +
lista2[3] - 1):
                            pog_m.append(lista)
                            pogodak = 1
                        else:
                            if (lista[0] == lista2[0]) and
(lista[1] >= lista2[1]) and (lista[1] <= lista2[1] +
lista2[3] - 1):
                                pog_m.append(lista)
                                pogodak = 1
                            if pogodak == 0:
                                red = "T"
            return [pog_t, pog_m]

def brodovi(n, m, d, tomek, medo, l):
    lista_pogodenih = provjera(l, tomek, medo)
    ucrtaj_brodove(tomek, medo, n, m, d)
    for lista in lista_pogodenih[1]:
        t.penup()
        t.forward(d * (lista[1] - 1))
        t.left(90)

```

```
t.forward(d * (lista[0] - 1))
t.pendown()
t.color('white')
t.begin_fill()
kvadrat(d)
t.end_fill()
t.color('black')
kvadrat(d)
iks(d)
t.penup()
t.back(d * (lista[0] - 1))
t.right(90)
t.back(d * (lista[1] - 1))
t.left(90)
t.penup()
t.forward((2 * n - 1) * d + 10)
t.right(180)
for lista in lista_pogodenih[0]:
    t.forward(d * (lista[0] - 1))
    t.left(90)
    t.forward(d * (lista[1] - 1))
    t.pendown()
    t.color('white')
    t.begin_fill()
    kvadrat(d)
    t.end_fill()
    t.color('black')
    kvadrat(d)
    iks(d)
    t.penup()
    t.back(d * (lista[1] - 1))
    t.right(90)
    t.back(d * (lista[0] - 1))
t.forward((2 * n - 1) * d + 10)
t.left(90)
return
```

## 2. Modifikacija teksta zadatka

Tomeku i Medi već su dosadile akcijske igrice pa su odlučili zaigrati igru Potapanje brodova. Iako je za nju dovoljno imati papir i olovku, Tomek i Medo su navikli igrati igre na računalu pa vas mole da im napravite simulaciju igre.

Napišite proceduru BRODOVI :n :m :d :tomek :medo :l koja će nacrtati mrežu od :m stupaca i 2\*:n redaka simetrično numeriranih kao na skici.

Donjih :n redaka predstavlja Tomekov, a gornjih :n redaka Medin dio ploče. Obojica najprije postave svoje brodove prikazane nizom uzastopnih kvadrata na pozicije zadane u listama :tomek i :medo.

Igru započinje Tomek. U svakom potezu igrač koji je na redu upita protivnika nalazi li se na polju [r s] dio njegovog broda. Ako je odgovor potvrđan, igrač koji je na redu je pogodio dio broda i ima pravo na još jedan potez. Ako nije, onda je na potezu protivnik.

Moguće je upitati protivnika više puta za isto polje — odgovor će svaki puta biti isti neovisno o broju upita.

Potrebno je prikazati stanje ploče nakon izvršavanja svih poteza iz liste :l, tako da se polja ploče predstave kvadratima stranice :d.

Unutar polja na kojima se nalaze pogođeni dijelovi broda potrebno je nacrtati dijagonale kvadrata, a polja na kojima se nalaze dijelovi broda koji nisu pogođeni potrebno je ispuniti crnom bojom. Gornji i donji dio ploče udaljeni su 10 piksela.

### Ulazni podaci

Brojevi :n, :m i :d su prirodni brojevi. Lista :l je prazna lista ili lista koja sadrži podliste oblika [r s], pri čemu r i s predstavljaju redak i stupac unutar protivničkog polja za koje je igrač koji je na potezu postavio pitanje.

Liste :tomek i :medo su prazne liste ili liste koje sadrže podliste oblika [r s], pri čemu su r i s brojevi koji predstavljaju redom redak i stupac unutar Tomekova, odnosno Medinog dijela ploče u kojem se nalazi dio broda.

**Komentar:** Iako je možda zamorno pisati puno listi, intuitivnije nam je birati svako polje zasebno za postavljanje broda, nego razmišljati o smjeru i duljini broda te kako će se to nacrtati ukoliko se odabere krivo polje kao početno.

### 3. Program - Potapanje brodova

#### Komentar:


Naš modificirani zadatak ne crta brodove različitih boja jer se na taj način olakšava igra. Protivnik bi znao koji je brod pogodio.

Program će na početku nacrtati dvije igraće ploče, jednu ispod druge.

Gornja igraća ploča je od jednog igrača, a donja od drugog. Igraća ploča ne prikazuje gdje se brodovi nalaze već samo služi za njihovo gađanje.

Igra kreće na način da svaki igrač u zasebnu tekstualnu datoteku upiše pozicije gdje će staviti svoje brodove.

Primjer zapisa u tekstualnoj datoteci:

 lada\_brodovi - Notepad

File Edit Format View Help

```
[3,1,"O",3],[3,4,"V",2],[2,5,"O",1]
```

Na taj način smo omogućili da igrači ne vide na koja polja je njihov protivnik pozicionirao svoje brodove.

Koordinate na ploči su kao i u antizadatku. Važno je napomenuti da za oba igrača vrijedi da se brodovi s:

- “V” protežu udesno, odnosno brod kreće od koordinate i zauzima određeni broj mjesta udesno
- “O” protežu prema gore, odnosno brod kreće od koordinate i zauzima određeni broj mjesta “prema gore” vizualno, dok bi možda intuitivnije bilo da se protežu prema sredini (što je drugačije u slučaju igrača čija je gornja ploča, no, ovo se lako ispravi zamjenom jedne left i jedne right naredbe)

Na nasumičan način program bira koji će igrač započeti igru (u našem programu će pisati imena Lada i Kevin).

Igrači upisuju protivničko polje koje žele pogoditi u obliku r, s, gdje je r redak, a s stupac polja na ploči. Ukoliko je igrač pogodio brod, program će ispisati “Pogodak” te će igrač ponovo upisati polje koje želi pogoditi.

Ukoliko je igrač promašio polje s brodom, program ispisuje “Promašaj” te je na redu sljedeći igrač. Kada su pogođeni svi brodovi jednog igrača, igra se završava te se ispisuje poruka tko je pobijedio.

## Funkcije:

`brodovi(n, m, d, dat_kev, dat_lada)`

- Funkcija simulira igru potapanje brodova tako da ju naizmjenično igraju dva igrača.
- Ulazni podaci:

<code>n</code>	broj kvadrata (polja) u retku
<code>m</code>	broj kvadrata (polja) u stupcu
<code>d</code>	duljina stranice kvadrata (polja)
<code>dat_kev</code>	datoteka s pozicijama brodova prvog igrača
<code>dat_lada</code>	datoteka s pozicijama brodova drugog igrača
- Funkcija vraća pobjednika igre

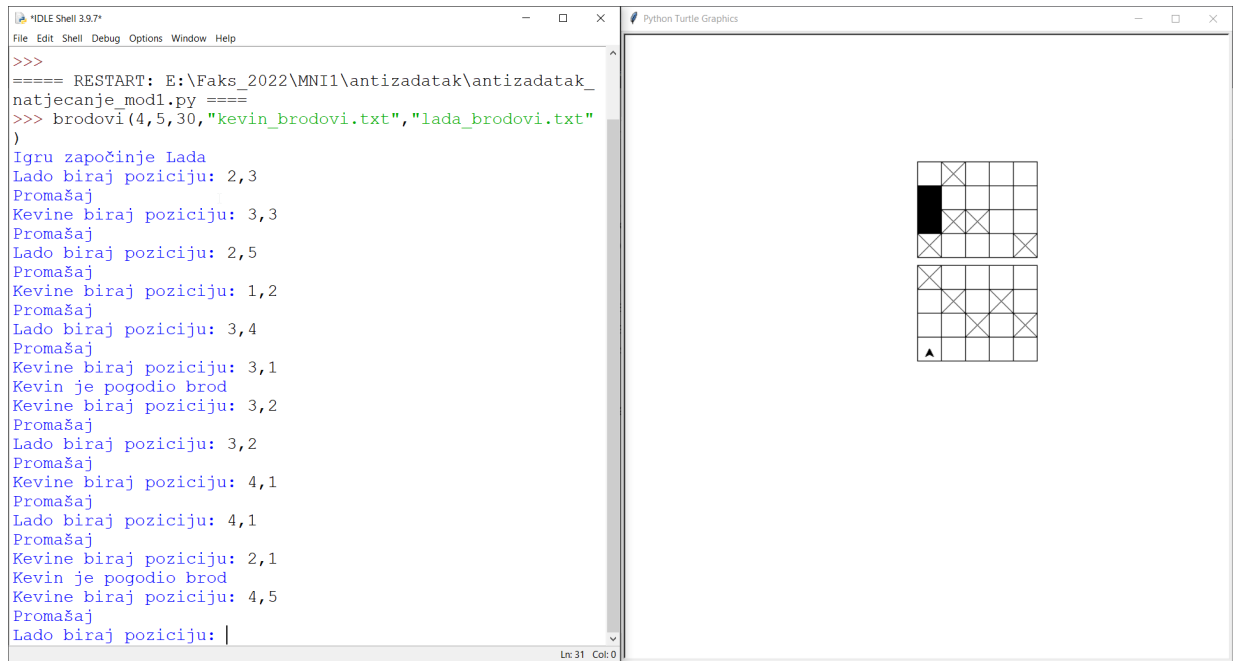
`ucrtaj_brod(potez, pokusaj, pogodak, n, d)`

- Funkcija ucrtava brod u mrežu ukoliko je pogođen, inače crta X
- Ulazni podaci:

<code>potez</code>	string oblika "K" ili "L" koji označava
	tko je na redu
<code>pokusaj</code>	koordinate polja koje se pogađa
<code>pogodak</code>	1 ako je pogođen brod, 0 inače
<code>n</code>	broj kvadrata (polja) u retku
<code>d</code>	duljina stranice kvadrata (polja)
- Funkcija nema izlaz

Ostale funkcije su kao i u zadatku (pogledaj gore).

Testni primjer:



## Kod:

```
import turtle
import math
import random

s = turtle.getscreen()
t = turtle.getturtle()
t.speed(20)

def brodovi(n, m, d, dat_kev, dat_lada):

    mreza(n, m, d) #crtamo mrežu
    t.left(90)
    #ovdje u listu upisujemo brodove iz datoteke
    #kevinovi brodovi dolje, ladini gore
    kev = open(dat_kev, 'r')
    line = kev.readline()
    kev_b = []
    mala = []
    for i in range(len(line)):
```

```
    if line[i].isdigit() == True:
        mala.append(int(line[i]))
    elif line[i].isupper() == True:
        mala.append(line[i])
    elif line[i] == "]":
        kev_b.append(mala)
        mala = []

lada = open(dat_lada, 'r')
line = lada.readline()
lada_b = []
for i in range(len(line)):
    if line[i].isdigit() == True:
        mala.append(int(line[i]))
    elif line[i].isupper() == True:
        mala.append(line[i])
    elif line[i] == "]":
        lada_b.append(mala)
        mala = []

#brojimo duljine brodova pa imamo counter
#da znamo kada igra staje
counter_k, counter_l = 0, 0
for brod in kev_b:
    counter_k += brod[-1]

for brod in lada_b:
    counter_l += brod[-1]

#Ovdje nasumično biramo tko prvi ide
novcic = random.randint(1, 2)
if novcic == 1:
    print("Igru započinje Lada")
    potez = "L"
else:
    print("Igru započinje Kevin")
    potez = "K"
```

```

pokusaji_lada, pokusaji_kevin = [], []
#ovdje kreće igra
while counter_k > 0 and counter_l > 0:
    if potez == "L":
        pokusaj = input("Lado biraj poziciju: ")
        if pokusaj in pokusaji_lada:
            print("Već ste pokušali ovu
poziciju. Pokušajte ponovo")
        else:
            pokusaji_lada.append(pokusaj)
            pogodak = 0
            for brod in kev_b:
                if brod[2] == "O":
                    if (int(pokusaj[2]) ==
brod[1]) and (int(pokusaj[0]) >= brod[0]) and
(int(pokusaj[0]) <= brod[0] + brod[3] - 1):
                        pogodak = 1
                    else:
                        if (int(pokusaj[0]) ==
brod[0]) and (int(pokusaj[2]) >= brod[1]) and
(int(pokusaj[2]) <= brod[1] + brod[3] - 1):
                            pogodak = 1
            if pogodak == 0:
                print("Promašaj")
                ucrtaj_brod(potez, pokusaj,
pogodak, n, d)
                potez = "K"
            else:
                print("Lada je pogodila brod")
                counter_k -= 1
                ucrtaj_brod(potez, pokusaj,
pogodak, n, d)
            else:
                pokusaj = input("Kevine biraj poziciju:
")
                if pokusaj in pokusaji_kevin:
                    print("Već ste pokušali ovu
poziciju. Pokušajte ponovo")

```



```

        else:
            pokusaji_kevin.append(pokusaj)
            pogodak = 0
            for brod in lada_b:
                if brod[2] == "0":
                    if (int(pokusaj[2]) ==
brod[1]) and (int(pokusaj[0]) <= brod[0]) and
(int(pokusaj[0]) >= brod[0] - brod[3] + 1):
                        pogodak = 1
                    else:
                        if (int(pokusaj[0]) ==
brod[0]) and (int(pokusaj[2]) >= brod[1]) and
(int(pokusaj[2]) <= brod[1] + brod[3] - 1):
                            pogodak = 1

            if pogodak == 0:
                print("Promašaj")
                ucrtaj_brod(potez, pokusaj,
pogodak, n, d)

                potez = "L"
            else:
                print("Kevin je pogodio brod")
                counter_l -= 1
                ucrtaj_brod(potez, pokusaj,
pogodak, n, d)

            if counter_k == 0:
                pobjednik = "Lada"
            elif counter_l == 0:
                pobjednik = "Kevin"
            return "Pobjednik je " + pobjednik

def ucrtaj_brod(potez, pokusaj, pogodak, n, d):
    t.penup()
    #t.left(90)
    if potez == "L":
        t.forward((int(pokusaj[0]) - 1) * d)
        t.right(90)
        t.forward((int(pokusaj[2]) - 1) * d)

```

```

    if pogodak == 1:
        t.begin_fill()
        kvadrat(d)
        t.end_fill()
        t.penup()
    else:
        iks(d)
        t.penup()
    t.back((int(pokusaj[2]) - 1) * d)
    t.left(90)
    t.back((int(pokusaj[0]) - 1) * d)
else:
    t.forward((2 * n - 1) * d + 10)
    t.right(180)
    t.forward((int(pokusaj[0]) - 1) * d)
    t.left(90)
    t.forward((int(pokusaj[2]) - 1) * d)
    if pogodak == 1:
        t.begin_fill()
        kvadrat(d)
        t.end_fill()
        t.penup()
    else:
        iks(d)
        t.penup()
    t.back((int(pokusaj[2]) - 1) * d)
    t.right(90)
    t.back((int(pokusaj[0]) - 1) * d)
    t.right(180)
    t.back((2 * n - 1) * d + 10)
return

def iks(a):
    dijagonala = a * math.sqrt(2)
    t.penup()
    t.forward(a / 2)
    t.left(90)
    t.forward(a / 2)

```

```
t.right(45)
t.pendown()
t.back(dijagonala)
t.penup()
t.right(45)
t.forward(a)
t.right(45)
t.pendown()
t.back(dijagonala)
t.penup()
t.right(45)
t.forward(a / 2)
t.left(90)
t.forward(a / 2)
return
```

```
def kvadrat(a):
    t.penup()
    t.forward(a / 2)
    t.pendown()
    t.right(90)
    for i in range(2):
        t.forward(a / 2)
        t.right(90)
        t.forward(a)
        t.right(90)
        t.forward(a / 2)
    t.left(90)
    t.penup()
    t.back(a / 2)
    t.pendown()
    return
```

```
def mreza(n, m, d):
    #n je broj redaka, m je broj stupaca, d je
    duljina stranice kvadrata
    for k in range(2):
        for j in range(n):
```

```
    for i in range(m):
        kvadrat(d)
        t.penup()
        t.forward(d)
        t.pendown()
    t.penup()
    t.back(m * d)
    t.left(90)
    t.forward(d)
    t.right(90)
    t.pendown()
t.penup()
t.left(90)
t.forward(10)
t.right(90)
t.pendown()
t.penup()
t.left(90)
t.back(2 * 10 + 2 * n * d)
t.right(90)
t.pendown()
return
```